Computer Programming Assignment 2024

Nidal Hevi Oğur – 3246336

This paper delves into applications of Simulated Annealing for solving K-SAT problems. It investigates the relationship between the number of variables N, and the algorithmic threshold of clauses, $M^{(Alg)}$, given that the empirical probability of solving a clause with simulated annealing is ½.

1. Problem Definition

We are given N binary variables $x = \{x_1,...,x_N\} \in \{0,1\}^N$ (where 0 represents **True** and 1 represents **False**) and M logical constraints -called clauses- each involving K variables from x, possibly negated.

To define a suitable cost function, we introduce the change of variable $\hat{x}_i = 1 - 2x_i$ for all $i \in \{1,...,N\}$ so that $\hat{x} \in \{-1,1\}^N$ (where 1 represents true, and -1 represents false). This way, negating a variable only requires multiplying it with -1. This allows us to describe the constraints as $s \in \{-1,1\}^{M \times K}$, where -1 means logical negation and 1 means no logical negation. We can now define the cost function as:

$$E(\hat{\mathbf{x}}) = \sum_{m=1}^{M} \left(\prod_{k=1}^{K} \frac{1 - s_{m_k} \hat{x}_{m_k}}{2} \right)$$

Where if $s_{mk}\hat{x}_{mk} = -1$ for every k the cost will increase by 1 and if $s_{mk}\hat{x}_{mk} = 1$ for some k the cost will not increase.

2. Simulated Annealing

In the Simulated Annealing scheme, the following parameters were used:

mcmc_steps: $10N$, anneal _steps: 500, $\beta_0 = 0.1$, $\beta_1 = -ln\left(1 - (0.95)^{1/10N}\right)$

Seed = 31.

These parameters are selected after careful consideration to find a balance between precision and running time. The parameter mcmc_steps was set to 10N for two reasons. Firstly, notice that each mcmc step proposes one move and there are N possible moves at each step -since each move proposes changing the sign of the literals that correspond to one of the N variables- thus the number of mcmc steps must be αN (i.e. proportional to N). While α should be large enough to consider enough possible moves so that at some point each variable will be proposed to change signs. It should also be small enough to keep the run time relatively short.

Such a value for α is determined to be 10, thus we have mcmc_steps = $10N$. Anneal_steps was set equal to 500 for similar reasons, to keep the run time relatively short and to ensure sufficient depth.

Before we delve the reasoning behind the selection of β_0 and β_1, first consider the following acceptance function:

$$P(\beta, \Delta c) = e^{-\beta \Delta c}$$

The acceptance function is defined as: Given the proposed move is increasing the cost, the probability of accepting that move ($\Delta c \geq 1$). The value of β_0 was set to 0.1 to allow the probability of accepting a move that increases the cost to be initially high, thus allowing the algorithm to be less greedy in the beginning, not allowing it to get stuck at local minima.

As we gradually increase β from β_0 to β_1, the algorithm will accept moves that increase the cost with lower probability (i.e. it will get greedier). To determine the value of β_1, we need to consider β such that the probability of accepting at least 1 move out of all the moves suggested is small enough so that we do not move away to a more costly position but also not so small that it would make the probability of accepting previous moves very small. Such a value is determined to be 0.05. We can now find such value for β_1 given the following assumptions:

- All proposed moves are bad moves (i.e. they increase the cost) since we assume the worst case.
- $\Delta c = 1$, since the probability of accepting a move when $\Delta c > 1$ will be even lower.

Following these assumptions, we have the following equation that represents the probability of accepting at least one move out of all bad moves proposed with a probability of 0.05.

If the probability of accepting a move is: $e^{-\beta}$ -since $\Delta c = 1$- then the probability of rejecting a move is $1 - e^{-\beta}$ and probability of rejecting all moves is $\left(1 - e^{-\beta}\right)^{10N}$ since we have mcmc_steps = 10N. Finally, the probability of accepting at least one move is 1-$P_{rejectAll} = 1 - \left(1 - e^{-\beta}\right)^{10N}$. Setting this probability equal to 0.05 and solving for β we have:

$$1 - \left(1 - e^{-\beta}\right)^{10N} = 0.05 \Rightarrow 0.95 = \left(1 - e^{-\beta}\right)^{10N} \Rightarrow \left(1 - e^{-\beta}\right)^{10N/10N} = 0.95^{1/10N}$$

$$\Rightarrow e^{-\beta} = 1 - (0.95)^{1/10N}$$

And finally, by taking logarithms, we have:

$$-\beta = ln\left(1 - (0.95)^{1/10N}\right) \Rightarrow \beta = -ln\left(1 - (0.95)^{1/10N}\right)$$

Thus, we set $\beta_1 = -ln\left(1 - (0.95)^{1/5N}\right)$. This decision for the value of β_1 is backed up by experimentation on the code with different values of β_1. Finally, the seed is chosen as 31 because it is the city license plate code of Hatay, the city the author is from in Türkiye.

2.1 Evolution of Acceptance rates

In this part of the investigation, we will look into the evolution of the acceptance rate during optimization i.e. we will observe the change in acceptance rates as we proceed in the annealing process. For this part of the investigation, the parameters were set to:

mcmc_steps: $10N = 2000$, anneal _steps: 150, $\beta_0 = 0.1$,

$\beta_1 = -ln\left(1 - (0.95)^{1/2000}\right) \approx 10.57$, Seed = 31.

The constant values of mcmc_steps and β_1 are due to the given constant N=200.

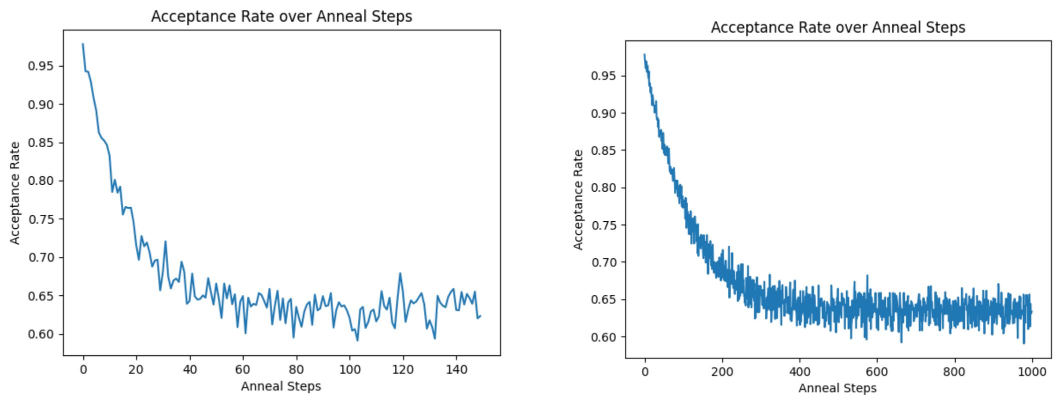


Figure 1: Evolution of acceptance rates during the annealing process with anneal_steps = 150 (left), and anneal_steps = 1000 (right)

The graphs above show the evolution of acceptance rates as we slowly increase the value of β. The graph in the right has higher annealing steps (i.e. a slower increase of the value of β) which increases precision. However, it is not very readable so we will examine the graph on the left which has lower annealing steps but is more readable and easier to analyse.

From the graphs, we can clearly see that the acceptance rate initially drops quickly as we increase β during the annealing process. This rapid change is due to the structure of the algorithm since we have a higher probability of accepting moves that increase the cost in the beginning. However, as we increase β, the probability of accepting a move that increases the cost, $P(\beta, \Delta c) = e^{-\beta \Delta c}$, is lower. The increase in the value of β will cause a decrease in acceptance rates as the algorithm becomes greedier over time. However, the rate at which the acceptance rate falls, decreases over time as the algorithm goes through the solution space. As the algorithm explores the solutions space, it will reach positions such that proposed moves are more likely to reduce the cost or keep it the same. Hence the decrease in acceptance rates will slow down gradually.

2.2 Can Simulated Annealing Always Find a Solution?

Repeating the same process with $M \in \{400,600,800,1000\}$ we observe that Simulated Annealing is not able to find a solution to every problem instance. Moreover, the number of problem instances for which Simulated Annealing is not able to find a solution increases with the value of M. As we increase the number of clauses M, with respect to the number of variables N, the algorithm will have a larger solution space and thus will be more likely to get stuck at local minima. We will see this more clearly as we investigate algorithmic thresholds in the next part.

3. Algorithmic Thresholds

The Algorithmic Threshold, $M^{(Alg)}$, is defined as follows. The number of clauses, M, for a given number of variables, N, such that the probability of finding a solution to the 3-SAT problem instance is approximately 0.5.

We are interested in finding the empirical probability of solving a random instance of 3-SAT problem. We define this probability $P(M, N)$ as:

$$P(M,N) = \frac{\#of\ solved\ problem\ instances}{\#\ total\ problem\ instances}$$

In our investigation for $M^{(Alg)}$ we will set the number of total problem instances equal to 30.

3.1 Probability of Finding a Solution for a Random 3-SAT Instance

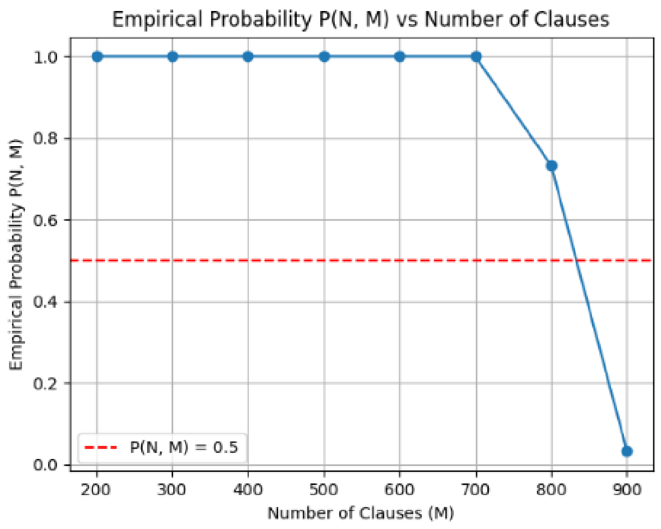


Figure 2: Evolution of $P(M, N)$ for N = 200, and increasing M.

After repeating the Simulated Annealing process 30 times for each value of M, we notice that $P(M, N)$ starts dropping after some point as we increase M. For these values of M, N, and number of total instances $M^{(Alg)}$ is around 836.36 (i.e. the probability of solving a random 3-SAT problem equals 0.5 when N=200 and M=836.36). The rapid drop in the value of $P(M, N)$ is due to the increase in the size of the solution space which increase the chance of the algorithm getting stuck at or around local minima.

3.2 The Relation Between $M^{(Alg)}$ and N

In this part of the investigation, we will look into how $M^{(Alg)}$ changes with respect to the choice of the number of variables, N. Repeating the annealing process for $N \in \{200,300,400,500,600\}$ we deduce that an increase in the number of variables will increase the algorithmic threshold. The following figure shows the relationship between $M^{(Alg)}$ and N.

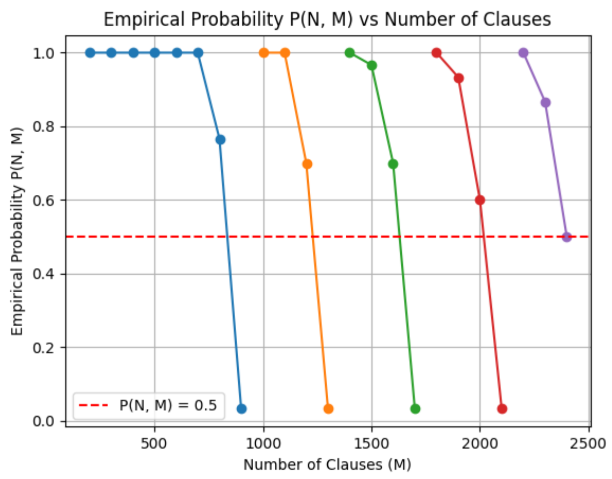


Figure 3: Evolution of $M^{(Algs)}$ for different values of N.

🟦 $N = 200$

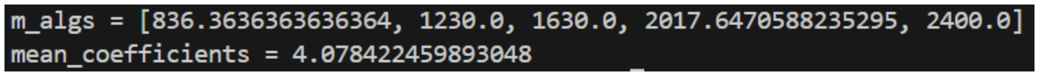
🟧 $N = 300$

🟩 $N = 400$

🟥 $N = 500$

🟪 $N = 600$

As can be seen from Figure 3, an increase in N increases $M^{(Alg)}$. The value of $M^{(Alg)}$ with respect to each N, rounded to the nearest multiple of 10 is as follows:


```
m_algs = [836.3636363636364, 1230.0, 1630.0, 2017.6470588235295, 2400.0]
mean_coefficients = 4.078422459893048
```

$M^{(Alg)} = 2400$ for $N = 600$, $M^{(Alg)} = 2017.647$ for $N = 500$, $M^{(Alg)} = 1630$ for $N = 400$

$M^{(Alg)} = 1230$ for $N = 300$, $M^{(Alg)} = 836.363$ for $N = 200$.

The relation can be more clearly seen from the following figures that shows the value of $M^{(Algs)}$ with respect to the values of N.

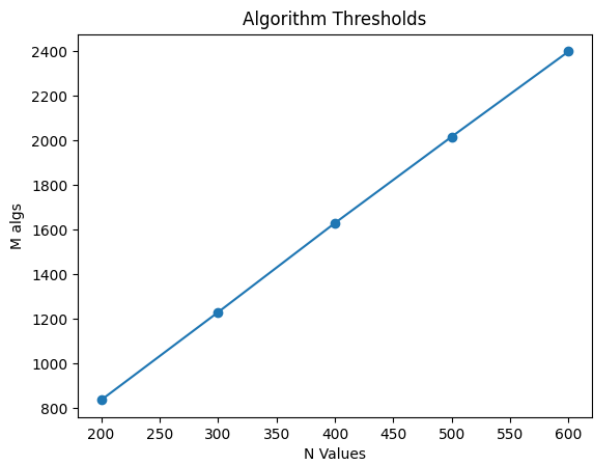


Figure 4: Values of $M^{(Alg)}$ for different values of N.

The figure suggests a linear relation between $M^{(Alg)}$ and N. More spesifically, each value of $M^{(Alg)}$ seems to be a multiple of the corresponding N value, we can write the equation $M^{(Alg)} = cN$. Where c is the coefficient of N that gives the corresponding value of $M^{(Alg)}$. We can find such c by dividing each $M^{(Alg)}$ with the corresponding N and taking the mean. Following this procedure we have $c \approx 4.078$. Thus we can conclude that $M^{(Alg)} \approx 4.078 \times N$.

Since we now know that the relation between $M^{(Alg)}$ and N is linear, when we plot $P(N, M)$ with respect to normalized M values, by removing the dependecy of M on N, we should expect the curves to look similar. We can do this by dividing each $M^{(Alg)}$ by it's corresponding N value and plotting $P(N, M)$ against these normalized M values.

The following figure shows collapsed curves of $P(N, M)$.

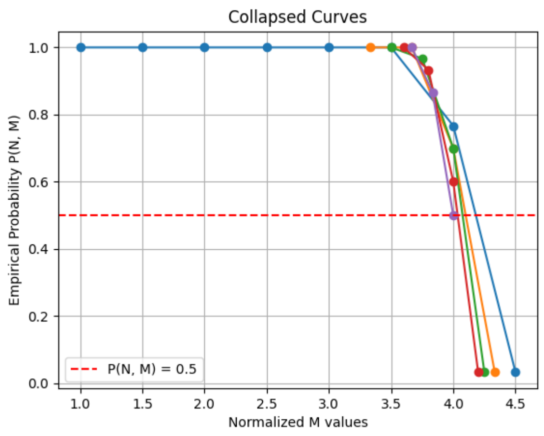


Figure 5: Value of $P(N, M)$ for normalized values of M.

🟦 $N = 200$

🟧 $N = 300$

🟩 $N = 400$

🟥 $N = 500$

🟪 $N = 600$

The graph in Figure 5 suggests that, by normalizing M (i.e. removing the dependecy on N), the curves behave very similarly, supporting previous findings. Moreover, each intercept of the Y=0.5 line is the corresponding coefficient, c, for that value of N.

Conclusion

This report explored the application of Simulated Annealing to solving K-SAT problems, focusing on the relationship between the number of variables NN and the algorithmic threshold $M^{(Alg)}$, where the empirical probability of solving a problem equals 0.5.

Some of the key findings are:

The algorithm successfully solves many problem instances but struggles as the number of clauses M increases relative to N, often getting stuck in local minima for larger solution spaces.

The acceptance rate decreases as β increases during annealing since we want the algorithm to be progressively greedier.

For N=200, the algorithmic threshold $M^{(Alg)}$ is approximately 836.363. The probability $P(M, N)$ drops rapidly beyond this point due to the increased difficulty of finding solutions in larger solution spaces.

The relationship between $M^{(Alg)}$ and N is approximately linear, with $M^{(Alg)} \approx 4.078 \times N$. This suggests that increasing N proportionally increases the algorithmic threshold $M^{(Alg)}$, maintaining a similar level of solvability.

Collapsing the curves of $P(N, M)$ for normalized values of M supports the linear relationship, confirming that the algorithmic threshold scales predictably with N.